PIP-II
Beam Pattern Generator Design/Status

John Dusatko
FNAL AD-ENG-RF-LLRF
April 13, 2020

Outline

- Introduction
- Purpose of BPG
- Description of its Function
- Design Approach
- Prototype vs Production
- PIP-II IT BPG (Proto)
- PIP-II BPG (Prod)
- Prod Version Design Ideas

Purpose of the BPG

- The PIP-II Beam Pattern Generator does the following:
  1) Generates signals that are used to select beam bunches in the Linac for injection into the Booster

  2) Generates an injection RF reference signal used by the booster for injection synchronization and bunch capture

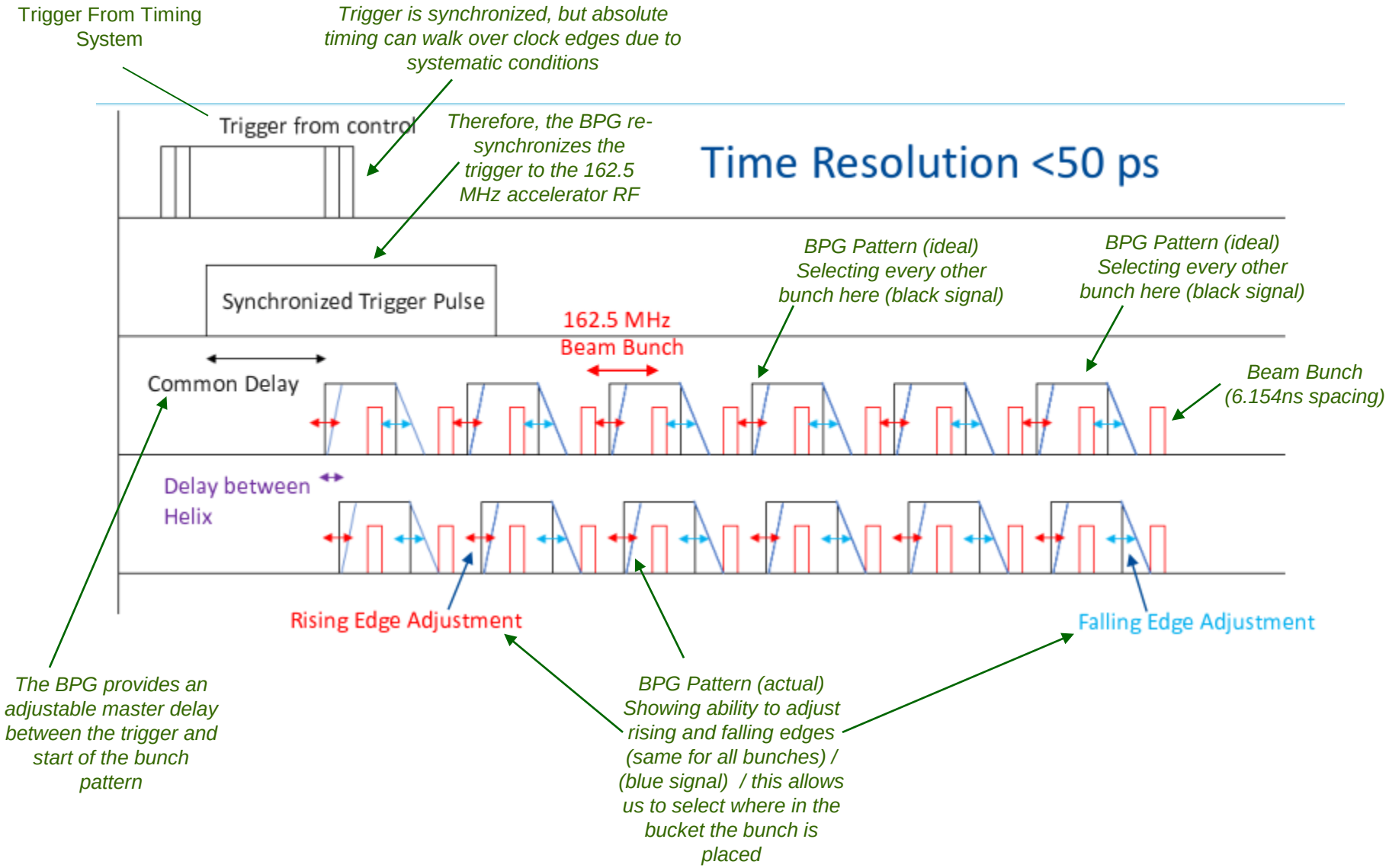
In more detail:

- Provides an adjustable, synchronized (to the accelerator RF clock) drive signal to the PIP-II LEBT and MEBT Kickers/Choppers (3 total, (1) LEBT, (2) MEBT) in a temporal pattern for bunch selection

- Provides an RF reference trigger the booster PLL for Linac → Booster bunch injection synchronization, as well as a revolution fiducial reset marker

- Reason: The LEBT, RFQ can generate 5mA beam (10mA max), but the rest of the Linac only has power to drive 2mA (2mA avg over 1us). In addition, different patterns of beam may be required by users. The BPG provides a flexible solution.

BPG Function – Linac Bunch Selection



BPG Function – Booster Sync

- Note to self: need to create timing diagram & block diagram for booster sync operation…

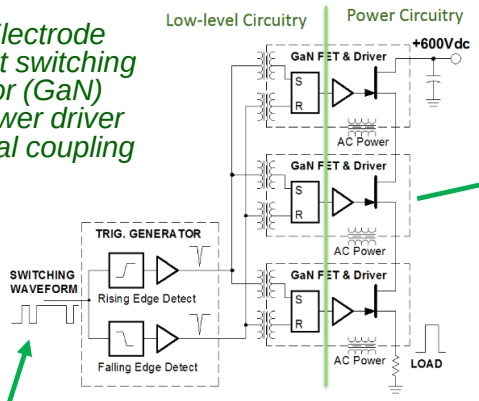
Beam Selection/Rejection

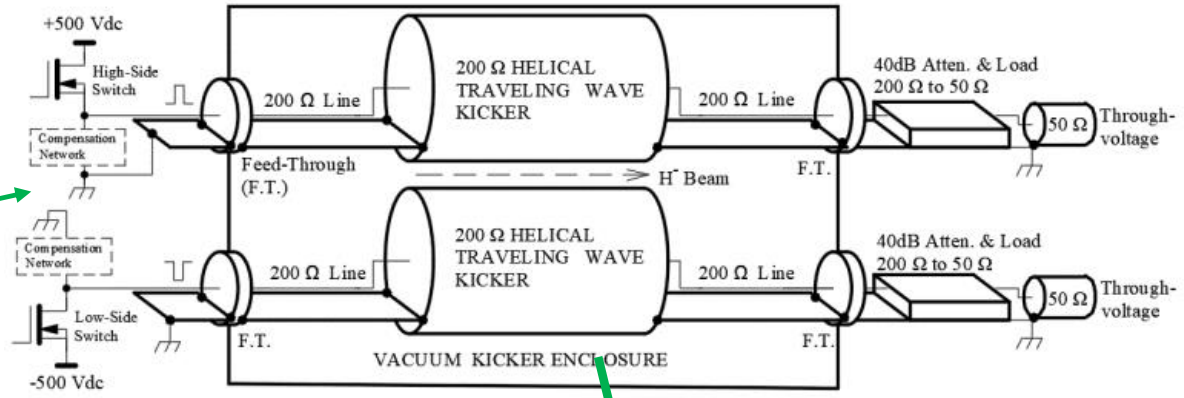
- HOW does this work?
  - A special chopper (kicker) structure was developed that allows the CW 162.5MHz Linac beam to be “chopped”
    - Beam buckets can be selected to either pass thru to the booster or be deflected into an absorber
    - There are three kickers:
      - (1) in LEBT
      - (2) in MEBT
  - This is accomplished using a special Z_0 = 200 ohm helical traveling wave kicker structure.
    - The kicker electrostatic wave matches the beam velocity.

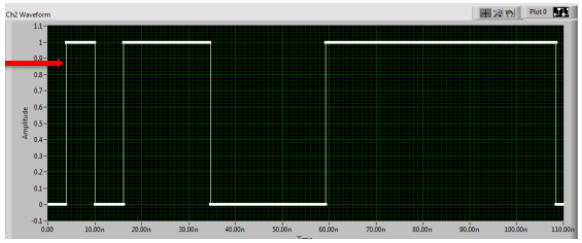
Beam Kicker Details

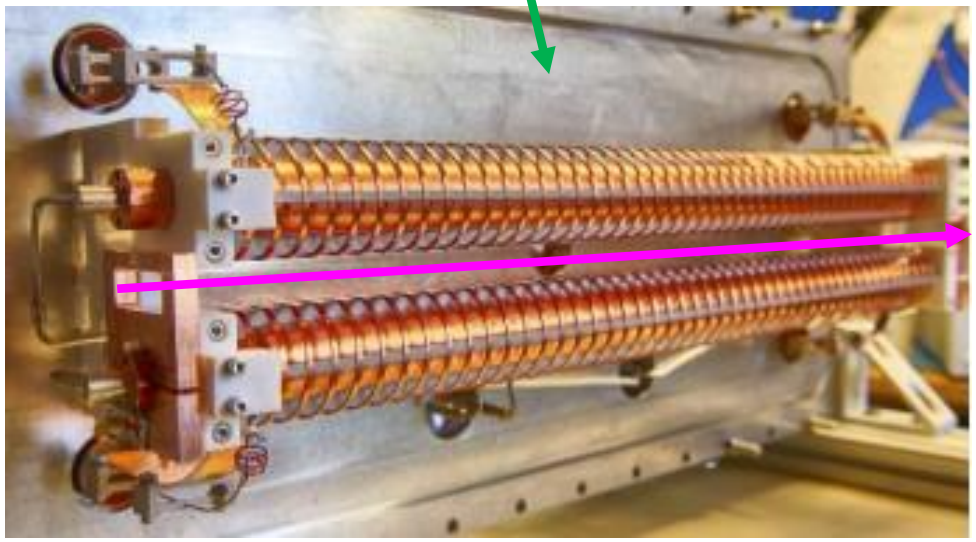
- Beam Chopper
  - Two helical kicker structures driven by a high voltage pulsed switch
  - Generates an E-field at +/- 500V potential

Kicker Electrode driver: Fast switching transistor (GaN) stack. Newer driver uses optical coupling







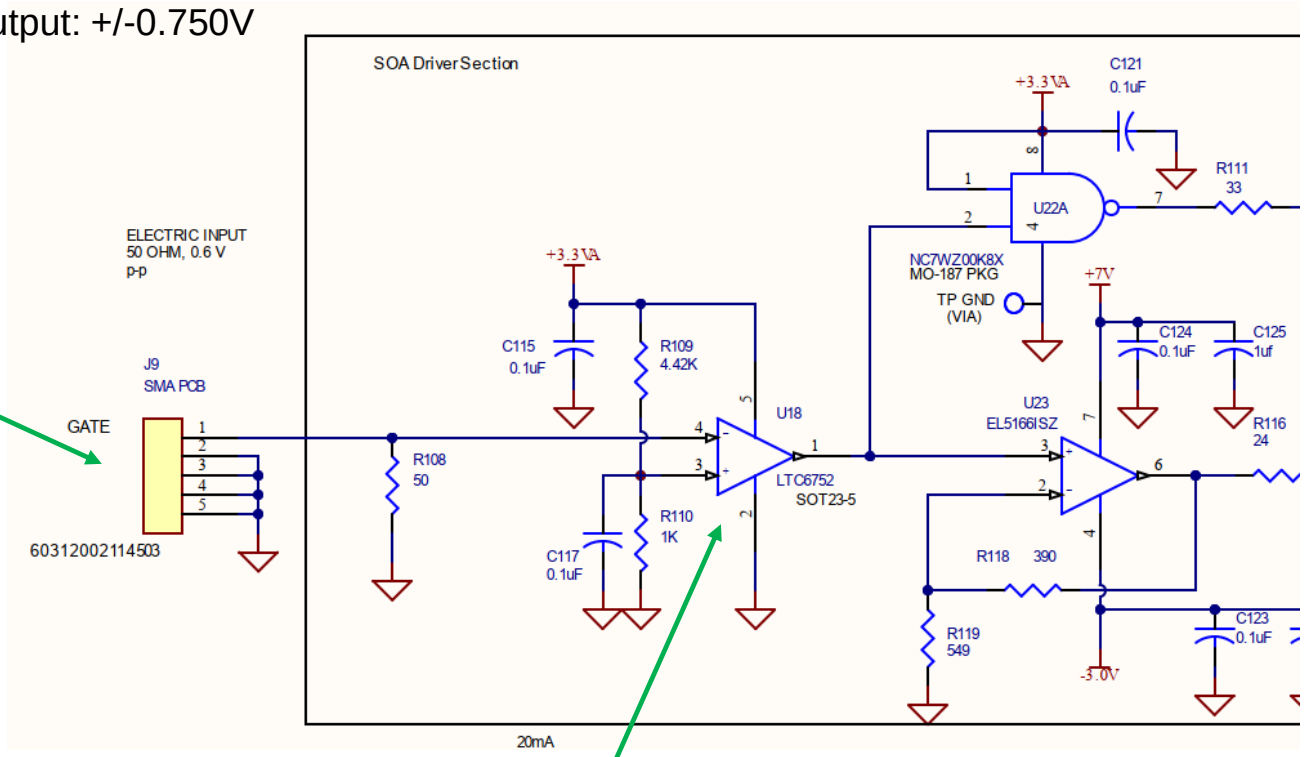


The BPG generates the waveform to the kicker drive electronics

Kicker photos: G. Saewert, et. al.

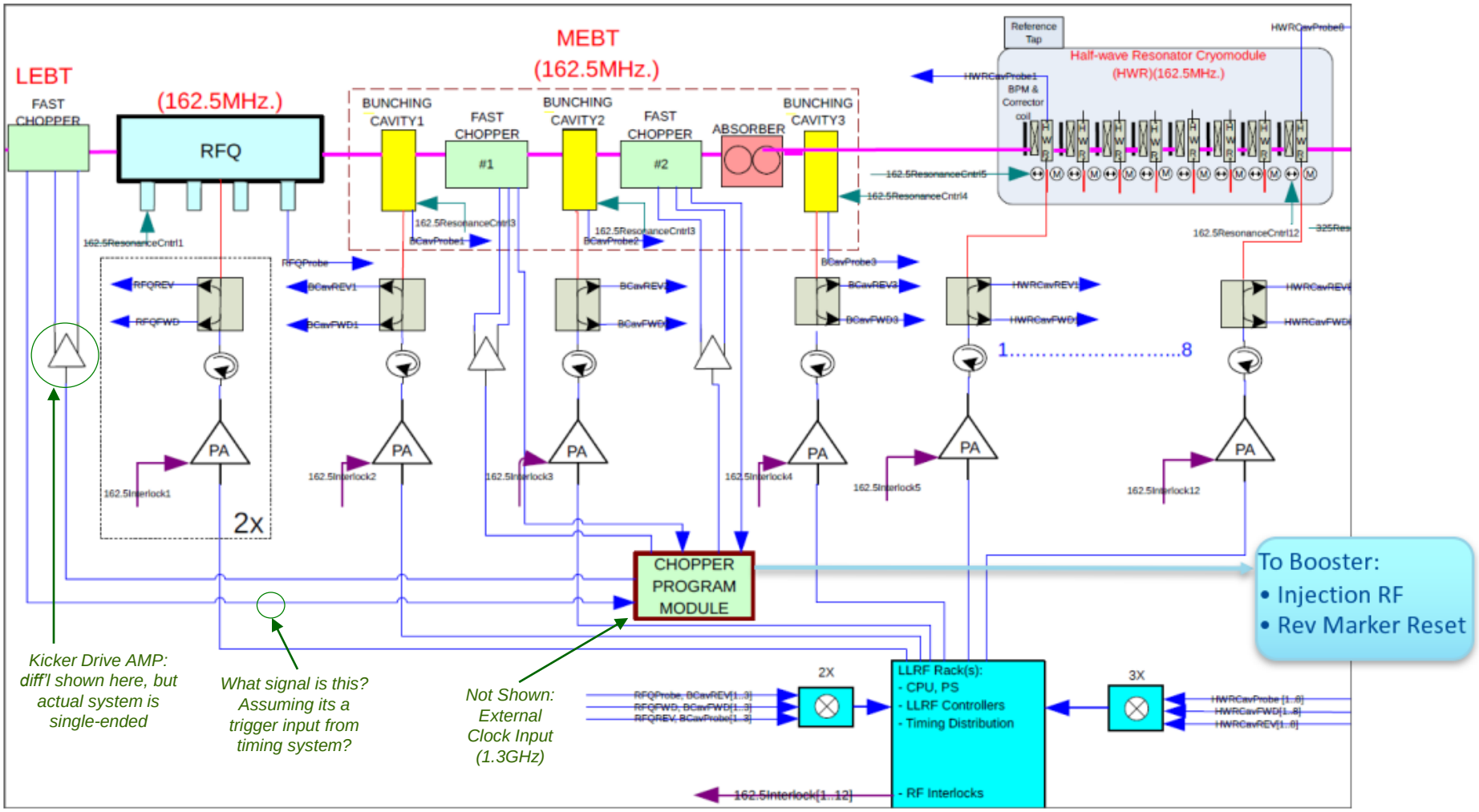
Beam Kicker Electrical Interface

- Kicker Electronics BPG Signal Receiver
  - Received by LTC6752 comparator (max toggle freq = 280MHz)
  - Threshold is set at 0.600V
  - Zin = 50ohms
  - Compatible with LVTTL
  - DAx2000 AWG output: +/-0.750V



How The BPG fits into the System

PIP2IT RF Station Diagram



Note: This diagram is conceptual and the interconnects may differ in the actual system. (e.g. the LEBT will most likely be driven by the MPS)

Design Approach

- The bunch pattern must be adjustable within each 6.154ns bucket → so need adjustable fine timing delay in the 10s of ps timescale

- The bunch pattern can be generated several ways:
  - Analog circuit: fast comparators and amplifiers with analog delay line
  - Discrete digital circuit: High-speed logic (ECL) coupled with fine digital delay lines
  - Arbitrary Waveform Generator (AWG) / Must be fast (sampling rate in GHz range)

➢ The AWG provides the most flexible/versatile solution:

Any desired bunch pattern and waveform type can be generated in software and downloaded and played out the AWG's DAC. Changes can even be made on the fly. Possible to even have adaptive waveform generation to compensate for systematic/environmental effects.

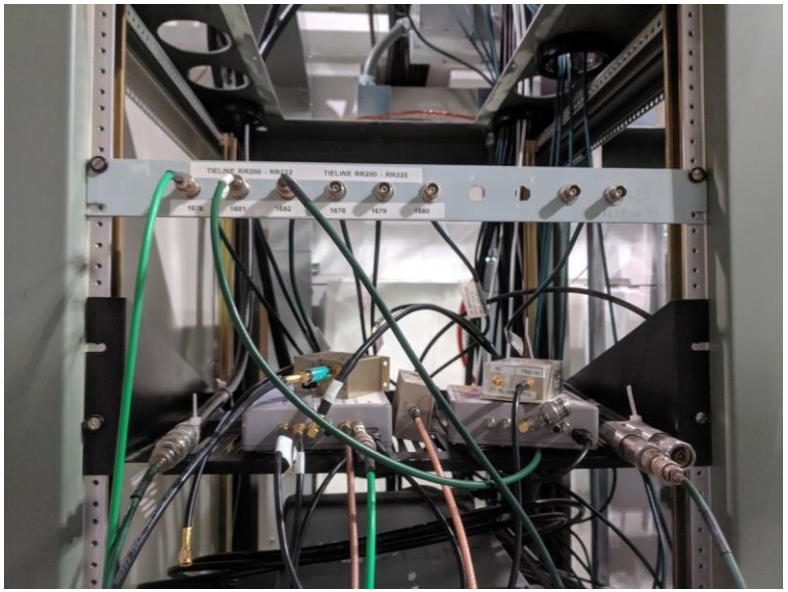
→ Therefore the AWG solution was chosen for this application

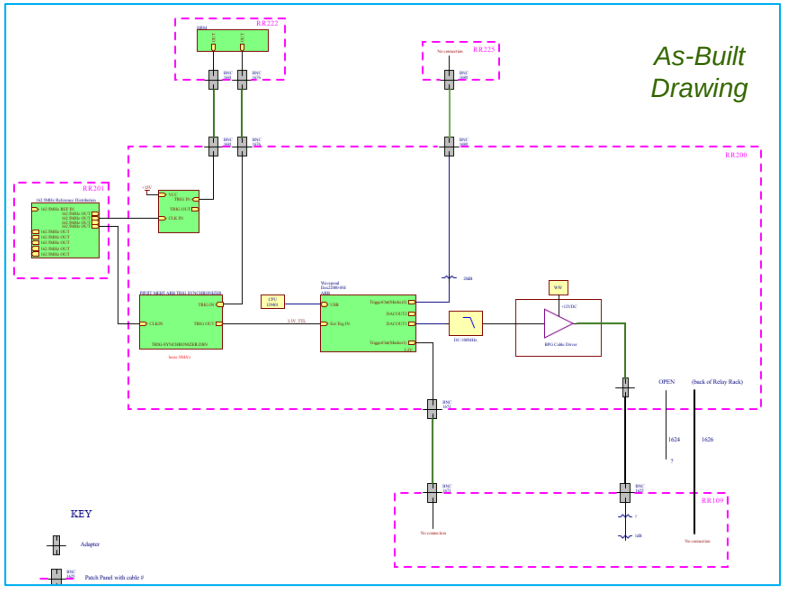
Prototype vs Production

- More specifically: **PIP-II IT** vs **PIP-II**
  - PIP-II IT BPG:
    - Rapidly assembled prototype: verified concept with single kicker
    - Not a fully integrated solution
      - AWG Required its own PC running labview connected over USB ifc
    - AWG is at end of life, mfgr no longer supports it, no replacement product (mfgr getting out of AWG business*) **they recently reversed their decision, but…*
    - Will keep using for PIP-II IT, but not a long-term solution for PIP-II
  - PIP-II BPG:
    - Need final production version
    - Fully integrated solution (control system, other PIP-II systems)
    - Should implement all requirements/features, room for growth
    - Could make core of the design general enough for use in other applications requiring high-speed DACs (maybe even add in high-speed ADCs)

PIP-II IT BPG System

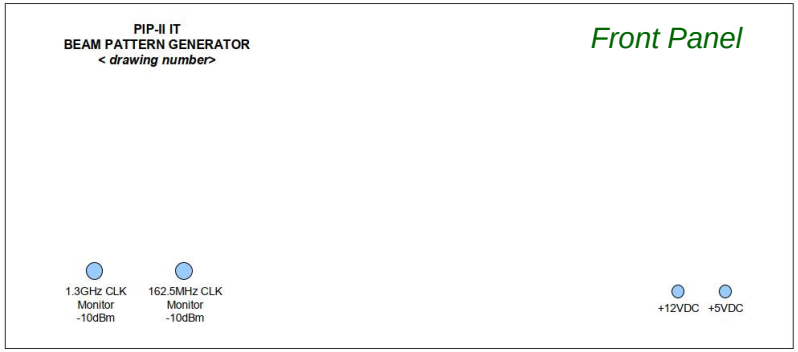
- **PIP-II IT Prototype System**
  - Built / Installed / Tested: 2017
  - Verified on single kicker
  - Things were “air-wired”
  - Remaining work:
    - Package into a proper chassis
    - Any design improvements:
      - Sync box – add 2nd DFF
      - Better driver circuit
    - “Own” System (Incl SW)
      - Need to be able to understand SW enough to operate & modify if needed
      - LabView

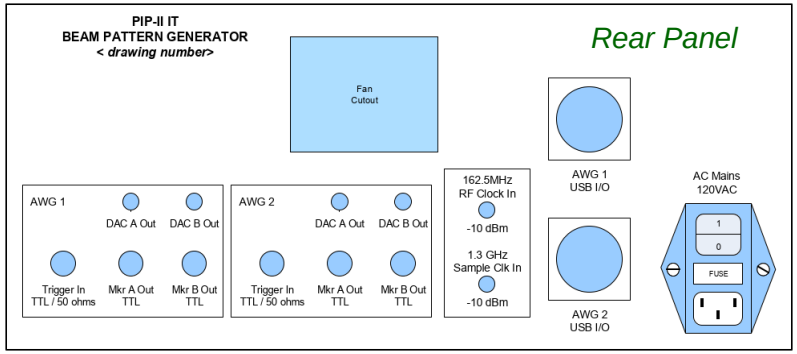




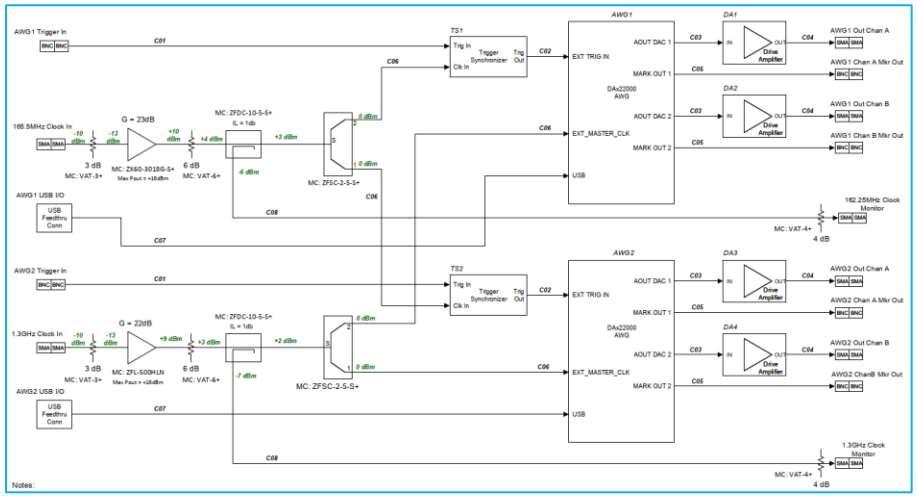
PIP-II IT BPG System

- **Packaging of PIP-II IT BPG:**
  - Components will be packaged into a 3U, 19-inch rack mount chassis
  - There will be four channels:
    - (2) MEBT Choppers
    - Booster Injection
    - Spare/LEBT
  - Use same LabView SW (with any necessary enhancements)

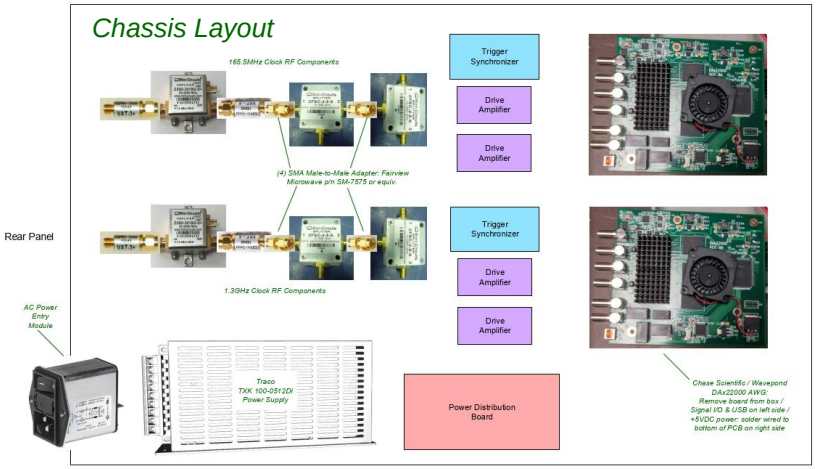




Signal Interconnect



Chassis Layout



PIP-II BPG Production Design

- For the Some ideas on how to approach this:

1) Outsource Design to company
   - Pros: Push off the design to someone else
   - Cons: Possible high NRE costs, don't own the design (unless we license it), previous experience has shown that this doesn't work very well (diverging goals)

2) Use existing boards
   - Pros: Leverage existing HW, we own everything, full control
   - Cons: Might not be able to implement all the features needed, and still need to design HW (but maybe not as much HW…)

3) Build our own
   - Pros: Full control of design, can implement exact solution, design could be used elsewhere
   - Cons: cost & schedule – both potentially higher

4) Have collaborator do so
   - Pros: Push off the design to someone else
   - Cons: schedule, engineering resources, reliable?, ownership issues? IP issues?

Design Approaches

- Some ideas on how to approach this:

1) Outsource Design to company
   - Pros: Push off the design to someone else
   - Cons: Possible high NRE costs, don't own the design (unless we license it), previous experience has shown that this doesn't work very well (diverging goals)

2) Use existing boards
   - Pros: Leverage existing HW, we own everything, full control
   - Cons: Might not be able to implement all the features needed, and still need to design HW (but maybe not as much HW…)

3) Build our own ← **Choose This Option**
   - Pros: Full control of design, can implement exact solution, design could be used elsewhere
   - Cons: cost & schedule – both potentially higher

4) Have collaborator do so
   - Pros: Push off the design to someone else
   - Cons: schedule, engineering resources, reliable?, ownership issues? IP issues?

Options For Build Our Own

- **For building it ourselves, here are some ways we could approach the design problem**:

  1) Full-Custom Board
  2) Use FPGA System on Module (SoM) with custom base board
  3) Use FPGA SoM with COTS FMC DAC/ADC boards
  4) Use Altera SoC Dev board with COTS FMS DAC/ADC boards

- Some Basic Design Assumptions:
  - Design will be a chassis-based, network attached device (NAD)
  - We will use the Altera Arria 10 SoC in order to leverage our existing FW & SW code base (as well as use existing dev tools…)
  - Will need at least (4) High-Speed (>/= 2.5GSa/s, 14b) DAC channels
  - Need Digital I/O for Trigger in / Marker Out
  - Would be nice to add some (4?) High-Speed (>/= 1.0GSa/s, 12b) ADC Channels
  - System I/O Will be Ethernet (1Gbit) / USB I/O optional (maybe for local console ifc…)
  - FPGA can be configured remotely

- The four options were explored, and a cost estimate was done for each one:
  - Considered cost
  - Pros/Cons
  - Looked at each option in technical detail
  - And ultimately decided on…

Options For Build Our Own

- **For building it ourselves, here are some ways we could approach the design problem**:

  1) Full-Custom Board
  2) Use FPGA System on Module (SoM) with custom base board ← **Choose This Option**
  3) Use FPGA SoM with COTS FMC DAC/ADC boards
  4) Use Altera SoC Dev board with COTS FMS DAC/ADC boards

- Some Basic Design Assumptions:
  - Design will be a chassis-based, network attached device (NAD)
  - We will use the Altera Arria 10 SoC in order to leverage our existing FW & SW code base (as well as use existing dev tools…)
  - Will need at least (4) High-Speed (>/= 2.5GSa/s, 14b) DAC channels
  - Need Digital I/O for Trigger in / Marker Out
  - Would be nice to add some (4?) High-Speed (>/= 1.0GSa/s, 12b) ADC Channels
  - System I/O Will be Ethernet (1Gbit) / USB I/O optional (maybe for local console ifc…)
  - FPGA can be configured remotely

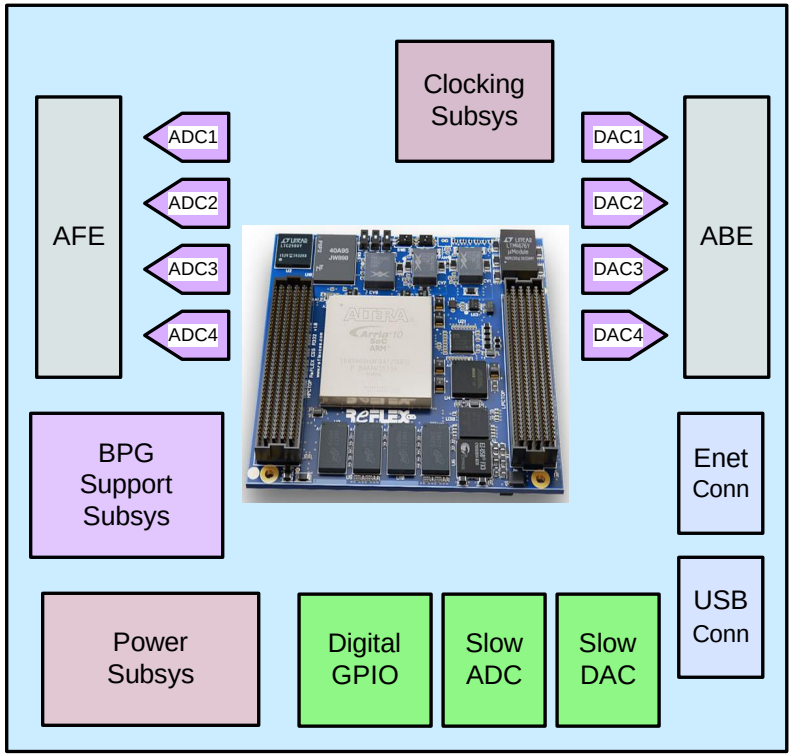
- The four options were explored, and a cost estimate was done for each one:
  - Considered cost
  - Pros/Cons
  - Looked at each option in technical detail
  - And ultimately decided on…

Option 2: Custom BaseBoard With FPGA SoM

- **Custom BaseBoard with SoM**
  - Baseboard contains data converters, clocking, power, I/O, but simplified
  - Use ReflexCES Arria 10 FPGA SoM board
  - DACs & ADCs on board
  - Analog Front/Back Ends can be configurable

Pros:
- PCB design is simplified, less components
- Less expensive than full-custom

Cons:
- Still developing complex PCB with
- Still considerable development time



Item	Time (hrs)	Cost
Labor:		
HW Development	**880**	**$88K**
FW Development	**320**	**$32K**
SW Development	**320**	**$32K**
Mech Development	**360**	**$48K**
System Test/Integ/Commis	**320**	**$32K**
	Total:	**$220K**
Materials:		
PCB	**--**	**$7.2K**
Components	**--**	**$27K**
Chassis	**--**	**$15.3K**
	Total:	**$49.5K**
	Sum Total:	**$269.5K**

Cost estimate is for Qty = 6 units

New BPG Design: BPG Features

- **BPG System** Features / Functions**:**
  - **Here is a working list of requirements (questions):**
    1) Resync trigger using 162.5MHz clock
    2) AWG sampling freq = 1.3GHz (does this need to variable as long as harmonic of 162.5MHz?)
    3) Delay controls: Global (trig to pattern start), Rising Edge, Falling Edge
    4) Programmable delay resolution = 38ps (fixed value?)
    5) Any special requirements for the marker signals?
    6) How many waveforms do we want to store? (drives how deep to make the AWG memory)
    7) How fast do new patterns need to be created?
    8) How fast do we need to switch between patterns?
    9) High Level Applications (HLAs): need to specify function – will this generate the patterns?
    10) How to distribute waveform/pattern information to other users?
    11) Interface and reaction to MPS

Cost estimate is for Qty = 6 units

New BPG Design

- **BPG** Features / Functions**:**
  - **Use ReflexCES Arria 10 FPGA SoM board**
    - Arria 10 FPGa
    - 4GB of DDR4 memory (for FPGA & HPS /ea)
    - FMC I/O
    - Plenty of comms peripherals (Enet, USB)
  - **DACs**:
    - (4) Channels: 2.8GSa/s, 16b (AD9144 or other…)
  - **ADCs**:
    - (4) Channels: 3.0GSa/s, 14b (AD9208 or other…)

    Note: DAC & ADC have JESD204B interface
  - **Analog Front/Back Ends can be configurable**: make them “raw” with pluggable daughterboards?
  - **General-Purpose Digital I/O**
    - 16/32 bits of bidirectional LVTTL
    - Handful of fast (ECL/NIM/LVPECL) I/O?
  - **General-Purpose Slow ADC & DAC**
    - ADC: (8) channels 1MSa/s 16b?
    - DAC: (8) channels 1MSa/s 16b?
  - ***Other features***?

Cost estimate is for Qty = 6 units

Further Thoughts

Thoughts / Questions / Comments:

1) Need to write up design documents:
   a) Functional Requirements
   b) Technical Specification
   c) Design Specification
2) Does the BPG need any sort of Machine protection interface / or another way to ask this: can the BPG but put into a state where it could cause damage (and if so, how do we mitigate the risk)?
3) Need to work out high-level user interface: how does one select the bunches? Right now we have a manually created .CSV file that selects the bunch pattern – any need to do this dynamically?
4) Still not clear on Booster interface: same signal as that going to the choppers?

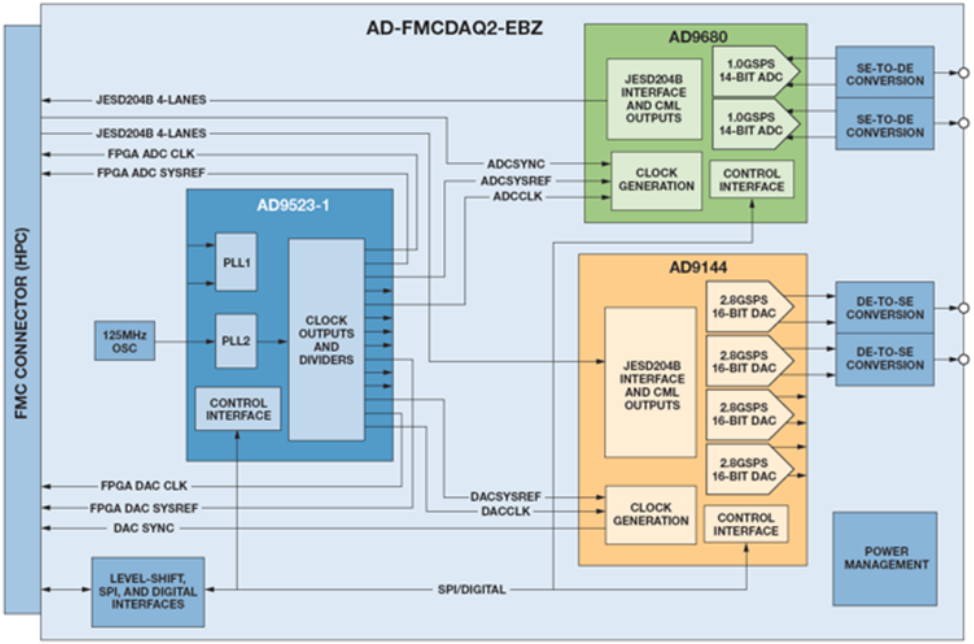
END

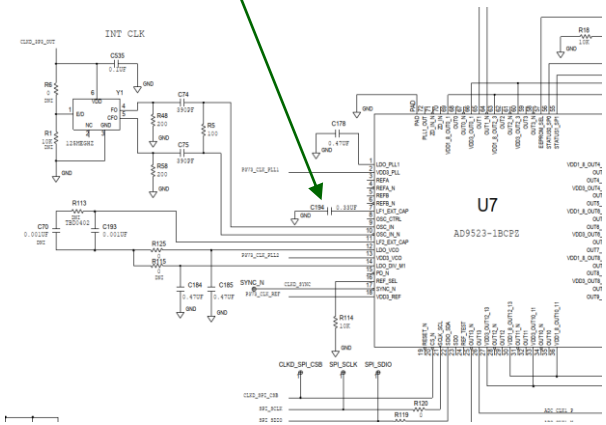
Supporting Slides

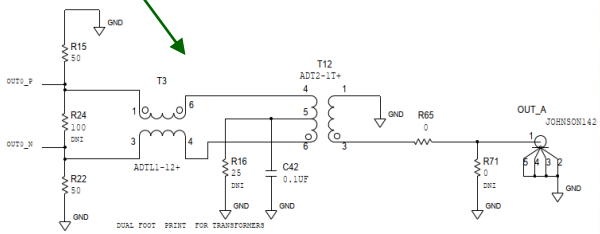
(mainly showing some FMC HW & other stuff)

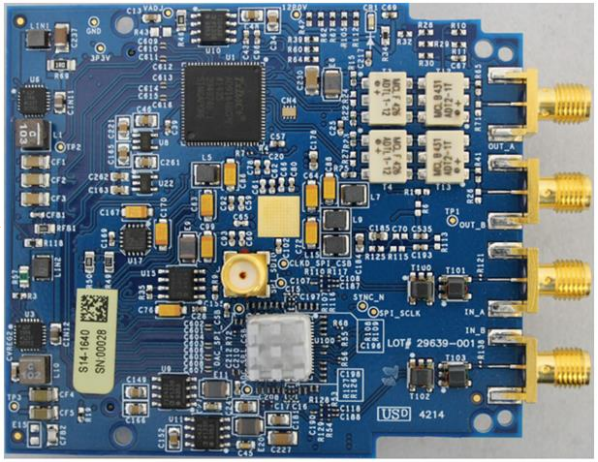
FMC-DAQ2-EBZ Details

- **Analog Devices FMC-DAQ2-EBZ**
  - $1500
  - FMC ADC/DAC: Analog Devices FMC-DAQ2-EBZ:
    - 2-Chan DAC: 2.8GSa/s, 16b
    - 2-Chan ADC: 1.0Gsa/s, 14b
  - Ext trigger input (SMA conn to LVDS level xlator)
  - Get all design source (sch, PCB, VHDL)
  - No easy Way get ext clock onto PCB without hacking it
  - DAC output is xfmr coupled, so we'd need to modify it as well…



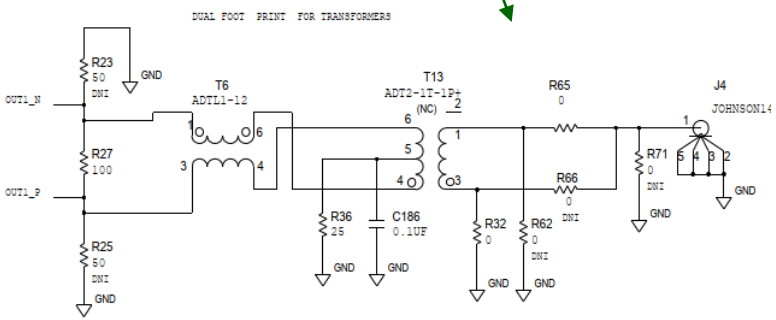


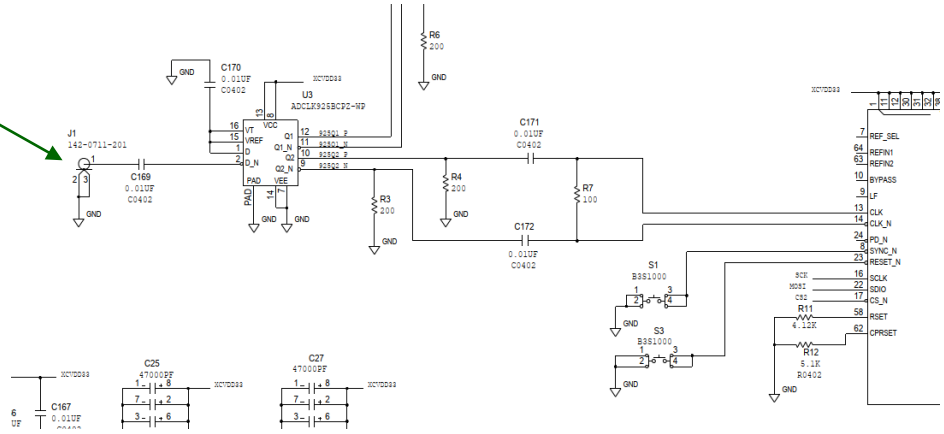




AD9144-FMC-EBZ Details

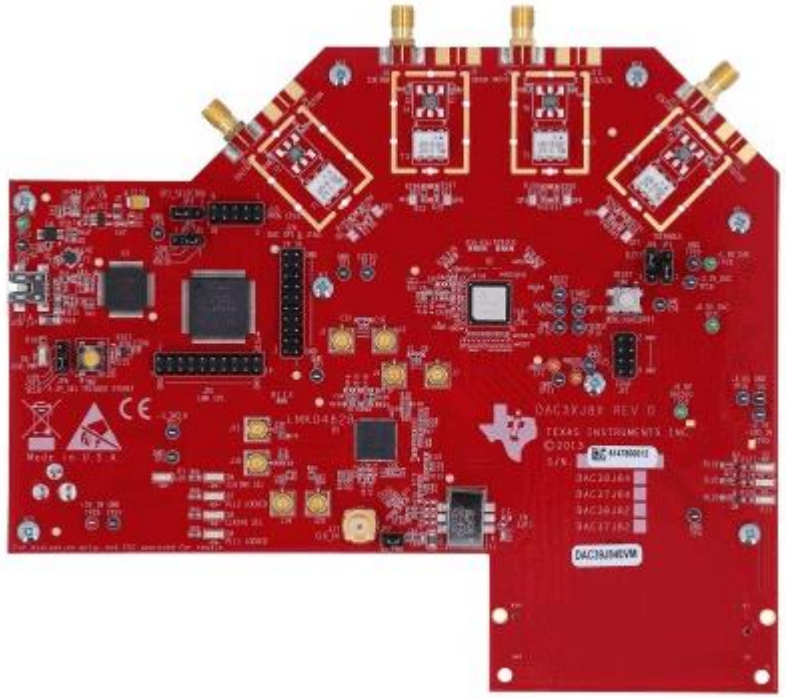
- **Analog Devices AD9144-FMC-EBZ**
  - $525
  - FMC-DAQ2-EBZ:
    - 4-Chan DAC: 2.8GSa/s, 16b
  - NO Ext trigger input
  - Get all design source (sch, PCB, VHDL)
  - Has Ext Clk input!
  - DAC output is xfmr coupled, so will need to hack board to make it DC-coupled

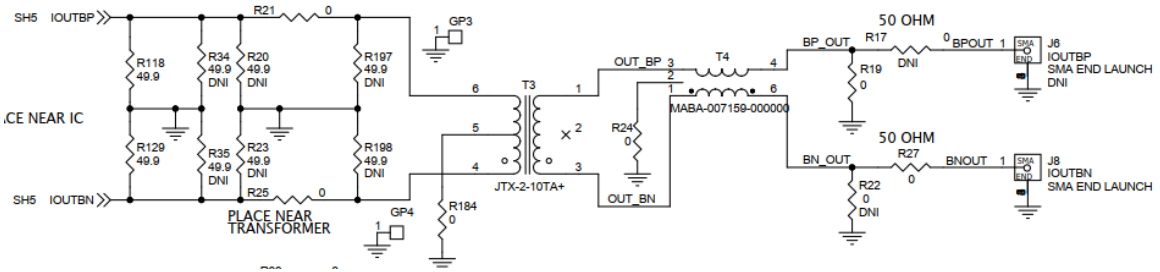


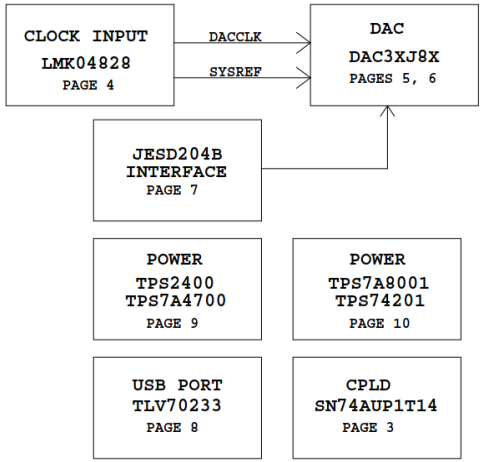


DAC39J84EVM Details

- **TI DAC39J84EVM**
  - $600
  - Eval Board:
    - DAC39J84 4-Chan DAC: 2.8GSa/s, 16b
  - Has Ext trigger input (sort of)
  - Get all design source (sch, PCB, VHDL)
  - Has Ext Clk input!
  - DAC output is xfmr coupled, so will need to hack board to make it DC-coupled







ReflexCES Achilies SoM BaseBoard

- **ReflexCES Arria 10 SoM BaseBoard**
  - $2500 for kit
  - SoM + Base
  - Has I/O for comms
  - But no Digital GPIO easily available…

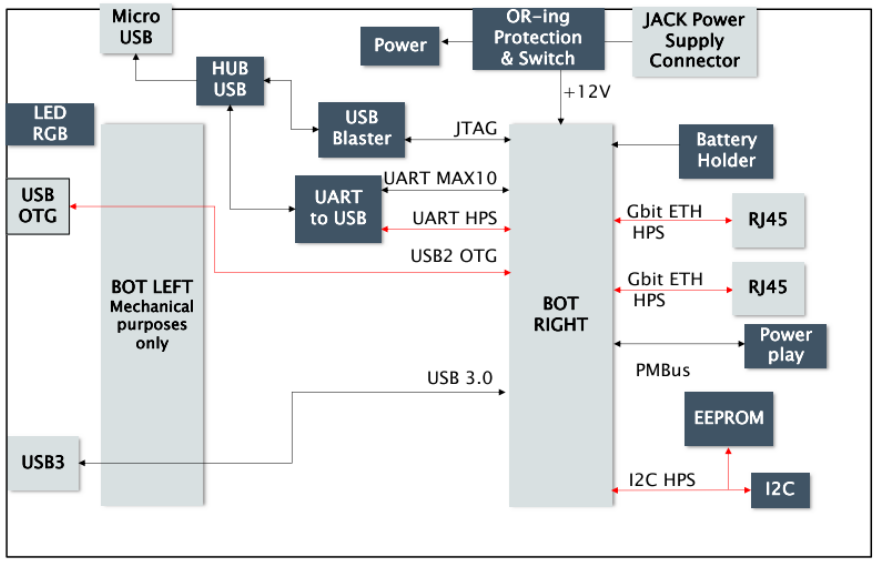


Figure 8: Achilles Starter Board Block Diagram